General 3D Acquisition and Tracking of Dot Targets on a Mars Rover Prototype

Todd E. Litwin, *Affiliate, IEEE*

***Abstract*— In June and July 2003 two Mars Exploration Rovers were launched to the Red Planet. Back on Earth, engineering-model rovers were driven on a mock Mars landscape in a large indoor sandbox. Characterizing their motion was accomplished by automatically acquiring dot targets mounted on their cluttered, upper surfaces from any position and orientation within the sandbox using a system of 12 ceiling-mounted cameras. A least-squares, *n*-camera, triangulation technique was used to attain typical 3D accuracies of 1-2cm within the 22m $\times$ 9m test area.**

***Index Terms*— Robotics, Vision, Acquisition, Tracking, 3D, Dot Targets, Mars Rover, Clutter, Occlusion, Triangulation, Least Squares, Camera Models.**

I. Introduction

On June 10 and July 7, 2003, the two Mars Exploration Rovers built by the Jet Propulsion Laboratory in Pasadena, California, were launched toward Mars from NASA's Kennedy Space Center in Florida. These rovers are the most sophisticated autonomous vehicles ever sent to another planet. They carry a number of instruments, including a suite of cameras [1] used for taking science and engineering images. A subset of the cameras were used by an on-board mobility system [2] [3] [4] [5] to create elevation maps of the terrain in order to avoid obstacles while navigating autonomously on the Martian surface.

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

The mobility system underwent extensive testing during development. Much of the testing with flight-like rover models took place on an indoors sandbox, measuring 22$\times$9 meters, and set up to simulate the Martian surface. Part of that testing required validating the motion of the rover as it drove across the sand. The visual tracking system described here was developed to provide the necessary ground-truth information.

Based on an earlier 4-camera monocular tracker created by the author for the 1997 Mars Pathfinder mission, the current system uses 12 cameras mounted approximately 5 meters above the sand to track a set of dot targets on the test rover's upper surface. It automatically acquires the initial positions of the dots wherever the rover may be within the sandbox, handling occlusions and discriminating between the dots and other visual clutter. It then tracks the dots as the rover moves. Each dot's position is determined using a multi-camera least-squares triangulation technique, the results being used to compute the position and orientation of the rover vehicle coordinate system within the sandbox.

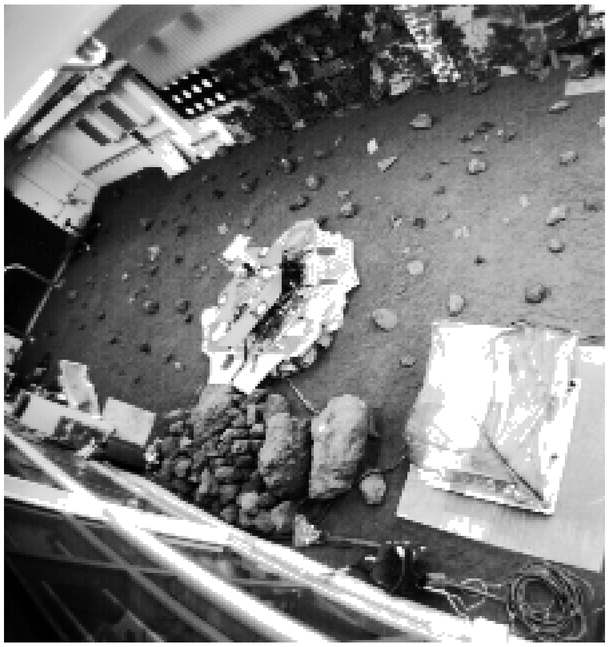

Fig. 1. Sandbox as viewed by one of the cameras. Note the rover in center, with target dots on solar panels, sitting on mock lander amidst much visual clutter.

II. Environment

The sandbox is a 22×9-meter region containing a mockup of the Martian surface. It is filled with a mixture of several types of reddish sand to mimic the appearance and texture of Mars. Volcanic rocks in a variety of sizes are scattered about as obstacles the rover must avoid.

In addition to these natural objects, a number of artificial ones are present. First there is the rover itself, along with a mock lander that has a visually rich set of air bags. There are building parts such as walls, doors, windows, air-conditioning ducts, and flourescent lighting fixtures. There is support equipment that includes raised flooring, large dots used for camera calibration, plastic protective covers, cabinets, tables, chairs, tool chests, hand tools, and other miscellaneous items. Sometimes there are people. All of these things fall in the fields of view of some or all of the cameras. Some of these items are specular.

Sometimes the scene is illuminated with standard flourescent lights. Sometimes it is illuminated with specially designed high-intensity lighting meant to mimic the color and intensity of the sun on Mars. While the cameras have lenses with hardware auto-iris support, the profile of image intensities does change when the lights do.

Amongst all this clutter the tracker must identify and track a 3-dot target on the rover's upper surface. It must do this even when other equipment mounted on the rover sometimes occludes dots from some camera views.

The only information provided about the dots is their diameters, their locations in rover vehicle coordinates, the range of distances they may be away from the cameras, and with what range of intensities they may show up in the images. Models describing how each camera views the scene are also available; see section III-D.

III. Algorithm

When the system starts, it automatically enters an acquisition mode, taking and analyzing images until it finds the rover. It then transitions into a tracking mode, where it repeatedly analyzes new images and updates the reported state of the rover. If it loses track for a minimum number of consecutive tracking cycles, it reenters acquisition mode.

A. Acquisition

The acquisition algorithm uses a *lazy-evaluation* approach. To manage the potentially explosive number of candidate solutions, it prunes away as many of those candidates as possible at each step before moving on to the next.

1) Capture: Sample new images from all cameras.

2) Identify: Find 2D dot candidates in each image individually. Start by segregating the image into pixels whose intensities indicate that they might or might not be dots. Find all 4-neighbor-connected regions of pixels that might be dots [6]. Eliminate regions that are outside an allowed range of 2D sizes derived from the expected range of distances from the camera.

3) Match: Correlate dot candidates between cameras, collecting a set of 3D dot finalists, each composed of a list of pairs of dot candidates. This is done by considering each pair-wise combination of regions between cameras to be a candidate dot pair.

Triangulate the two views of each region's 2D centroid to find closest crossover point of the two lines of sight. Eliminate the pair if the triangulation is colinear or intersects behind the camera, if the distance is outside the expected range from either camera, if the distance between the lines at the point of minimum cross-over is larger than the diameter of the largest target dot, or if either region is outside the allowed range of 2D sizes for the computed distance from its camera. Project the mean cross-over point back into each camera and eliminate the pair if the projection does not fall within the bounding box of either camera's region. If it survives all tests so far, then accept the current candidate pair as ready to join the ranks of finalists.

Look through all finalists already identified. Find the closest prior finalist whose projection from 3D into the 2D for each of the candidate pair's views falls within the bounding box of the pair's 2D regions in both cameras. If a prior finalist is identified, then merge the new candidate pair into a list of such pairs maintained for each finalist, updating its 3D position as computed over all of its pairs, computed for speed as the mean of each pair's triangulated mean point of nearest cross-over. If a prior finalist not identified, then create a new one containing only the new candidate pair.

4) Select: If there are fewer than *n* finalists, abort and try again. Otherwise choose the *n* dots best matching the target array. Start by recomputing each finalist's 3D position using a least-squares calculation of the best cross-over point of the set of all lines, weighted by the inverse-square distance from each camera; see the appendix.

Consider each subset of *n* dot finalists individually from the whole collection. Eliminate any subset that is upside down to handle cases with mirror symmetry about a horizontal axis. Compute the variance of the *n* 3D dot positions and eliminate any subset whose variance is more than twice the variance of the set.

Examine each permutation [7] individually of each surviving subset. Compute a score for each permutation based on how well it matches the dot sizes and inter-dot distances of the target. Compute the score as the square of the differences of the inter-dot distances for the actual and target dots, plus the square of the differences of the dot diameters for the actual and target dots.

Choose the best permutation of all subsets. Check that it has a score close enough to an ideal target to be reasonable. If not, abort and try again.

Report the 3D position of each dot.

5) Locate: Compute the state of the target. First compute the position and orientation of the target based on the 3D positions of each dot, using the quaternion method of Hebert [8]. Then transform the ideal dot positions with the position and orientation, comparing them to the measured positions. If the RMS residual is over a threshold, reject the solution. Finally, run a single iteration of the tracking algorithm on the result, rejecting the acquisition if that algorithm doesn't like it.

B. Tracking

1) Capture: Sample new images only from cameras that can see the target's previous 3D position.

2) Update: Find each dot in each camera. First project each dot's previous 3D position into 2D for each camera. Consider a bounding box around its projection that is 3 dot radii away in all four directions. Find 4-neighbor-connected regions as described earlier. Eliminate a region if it is outside the allowed range of 2D sizes for the computed distance from its camera Find the closest region to the previous 3D position's 2D projection. Eliminate region if it is more than 1.5 dot radii away since the tracker runs quickly enough that we do not expect this to be reasonable. If any two of the dots in one camera view select the same region, abort and try again.

3) Triangulate: Compute the 3D position of each dot using the weighted least-squares method described earlier. If any dot has moved more than an input maximum distance, abort and try again.

4) Locate: Compute the state of the rover. Compute the position and orientation of the target based on the 3D positions of each dot as described earlier. Transform the ideal dot positions from rover to world coordinates using the new position and orientation, and compare to measured positions. If the RMS residual is over a threshold, reject the solution.

C. Areas for Improvement

There are a number of possible refinements that could be explored. Here are some thoughts on the base algorithm:

- At one point in the acquisition algorithm a search is made for the closest prior finalist that matches a new one. This is currently done by projecting the prior one's current 3D position into each candidate pair's image to see if it falls within the bounding box of the region. Sometimes this test is too stringent, leading to a splintering of sets. It might be reasonable to extend the test into 3D by looking at range uncertainties, or else by enlarging the 2D bounding box in both views using projections or range windows.
- The tracker is nearly stateless. While it uses the past 3D dot solutions to seed the next cycle, those values are forgotten thereafter. In the face of noise there is some jitter in the solutions. A Kalman or other filter could be applied to the dot positions, although it would be important to control the parameters to prevent undue lag in response to sudden movements.

Efficiency is also an important consideration. For example, only 3-dot targets are used at present even though the system is set up to support configuration for any reasonable number of targets greater than 2. Extending to 4 dots already raises the number of combinations so high that acquisition goes from a few seconds to several minutes. Anything that could be done to eliminate bad candidates earlier in the process or otherwise speed things up should be considered.

- When the current algorithm is looking at the size of a dot in an image, it only considers the larger dimension of its bounding box. Since this is an overestimate in most cases, the tests are not as stringent as they might be. Using the covariance approach to estimate the semimajor axis of the dot's elliptical projection into the image would give a better estimate of its size and allow bad matches to be thrown out earlier.
- There is one place where the smaller dimension of the dot's bounding box is used in a test. The

covariance approach's calculation of the semiminor axis would be an improvement here, too.

- The semimjor and semiminor axes could be used together if we knew what angle the dot presented to the camera. The current system does not provide dot orientation knowledge as input, only position. If the system were upgraded to take in a normal vector to the dot's surface, the elliptical shape seen might be used to advantage. It is easy to see that the tracking cycle could use this as a check after determining the full 3D pose of the rover. While it would be harder to use during acquisition, finding a way to do so could improve robustness and speed significantly, and would be worth exploring.
- The ellipticity of a true dot's projection into the image could be applied yet another way. A detailed check of the symmetry of the shape could be used early in the acquisition algorithm to filter out shapes that do not look at all like ellipses.

As mentioned earlier, the tracker's cameras have auto-iris hardware to control exposure. There is also an unused ability to have software take control of the iris setting. This might be used to overcome the occasional saturation of CCD pixels that causes blooming. Such blooming, which occurs for some dot orientations and lighting conditions, causes an asymmetrical enlargement of the dots, distorting the computed dot centroid. In extreme cases the dots can be completely obscured by this effect.

D. Camera Models

It is important to note how critical it is to have high quality camera models. It is obvious that without the ability to project between 2D image and 3D world coordinates it would be impossible for any acquisition or tracking algorithm to function. But it is even more important to have very accurate models when trying to perform the kind of sensitive geometrical reasoning that the acquisition algorithm described here does. Since a number of the pruning steps depend upon using partial derivatives of the 2D-to-3D and 3D-to-2D projections, the formalism used to represent camera models must provide this as well. For a real-time system, of course, performance is also a desirable feature.

We are fortunate in the JPL robotics program to have a family of camera models that satisfies all these requirements. This family starts with a linear model developed at JPL in the late 1970s [9] and used extensively in the research program until the early 1990s [10]. With a compact 4-vector representation, it is very efficient. It takes only 2 dot products, 2 subtractions, and one division to project a 3D point into each 2D coordinate, for example.

The rover tracker uses the next member of this camera-model family [11], which adds radial lens distortion to the linear model, a feature typically needed to retain high accuracy as fields of view increase. A final member of the family [12], not needed for the tracker's cameras, adds a moving entrance pupil and generalizes the earlier models to include both perspective-projection and fish-eye cameras along with other optical geometries.

These models were used at JPL only by research programs through the mid 1990s. They then started to see service in flight projects. They debuted with the Mars Pathfinder IMP cameras [13], and are currently in use for all the flight cameras on the Mars Exploration Rovers [1]. It is expected that they will be used in future missions as well.

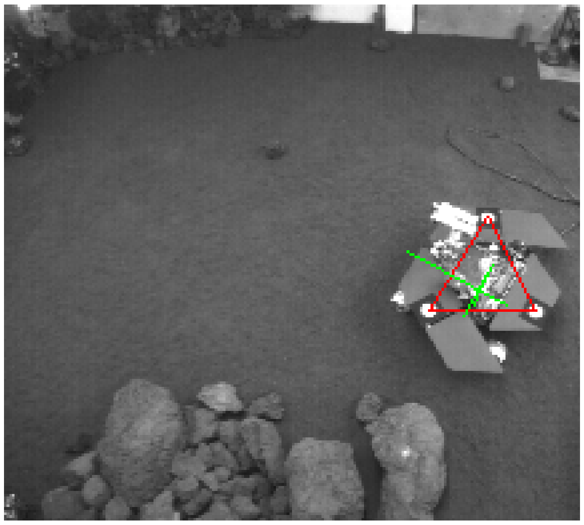

Fig. 2. Display of tracker in action. Red lines connect projections of located dot positions. Green lines form coordinate axes with elongation in the forward direction.

IV. Performance

The tracking system runs on a 350MHz PowerPC processor, controlling PMC-based image-acquisition and graphics-display modules and a VME A/D board. It usually takes a few seconds to acquire the rover's position and orientation. Tracking generally runs at about 6Hz, depending upon how many cameras can see the dots at any given time.

The primary output of the tracker is the position and orientation of the rover vehicle coordinate system, along with a time tag. This is augmented by information on the dot locations, both in 3D and in 2D, and error estimates.

To validate the tracker's absolute accuracy, a dot target was placed in several locations in the sandbox. The tracker's output was compared to that of a hand survey. The discrepancy between the two measurements was 1.5cm $\pm$ 0.6cm over approximately 18 meters.

To validate the tracker's relative accuracy, the output data was logged for a series of runs in which a dot target was moved in approximately straight lines across the sandbox, with both longitudinal and transverse runs. This was accomplished by suspending the target by a rod from a large ceiling crane normally used to move equipment around the sandbox, and then moving the crane in one dimension at a time, covering almost the entire sandbox. There was some lateral vibration of the dangling target visible, although it appeared to be slight. The vibration, as well as any non-linearity of the crane rails, certainly affected the results, which must be considered upper limits to the system's true accuracy. Those results showed that the RMS deviation from straight lines was 8mm $\pm$ 2mm horizontally and 6mm $\pm$ 3mm vertically across all the runs, with the maximum observed RMS deviation of a single run being 12mm. The maximum deviation for a single measurement across all the runs was 56mm.

While these results are quite satisfactory, there were times when the system didn't work so well. Very occasionally the tracker would lock onto the wrong object, producing wildly wrong results. There were also degenerate cases where partial occlusion would cause unusually large errors in the dot centroids. The results in the previous paragraph have a few such cases removed.

Some of the potential algorithm improvements discussed in section III-C would have eliminated many of these problem cases. Better surveying of the dots also would have helped. When the surveying was poor, some of the parameters controlling tolerance to error were loosened to allow successful acquisition to take place. This also made the system more prone to accepting bad data as good.

Very rarely the rover would be sitting in such a position such that no dot target was visible by at least two cameras without occlusions. This defeated the acquisition algorithm, leaving the system repeatedly trying to acquire, and making it more prone to locking on to the wrong target. If the rover passed through such poses briefly while moving, the tracking algorithm was usually able to bridge the gap.

V. Conclusion

The work here shows one promising approach to acquiring and tracking a dot target in a relatively unconstrained environment, with pointers to areas for further work. It was used to log almost all test runs for the mobility system. It was used similarly for testing the rover's low-level attitude-determination system. It was also used to initialize the state of the vehicle prior to many test runs.

Overall the tracking system performed very well. It was generally fast, accurate, and resilient to clutter. Its automatic acquisition and usually robust tracking contributed greatly to user satisfaction, making it easy and reliable enough to be used often. While it did have some occasional problems, in the end it provided much needed information to the testers, and contributed to mission readiness.

Appendix
Minimize Squared Distance of Point to a Set of Lines: "Best" Intersection or Crossover of Lines

The following presents the derivation for the equations used to compute the best 3D point for a target implied by projecting the 2D image coordinates out as rays from *n* cameras. Since in general these rays do not intersect, a least squares calculation is made for the point that minimizes the sum of the squared perpendicular distances from those rays.

Let $\vec{b}$ be a base point and $\hat{u}$ be a unit vector defining one of the lines, as shown in Fig. 3. Let $\vec{p}$ be the target point "near" the line. Let $\vec{d}$ be the perpendicular difference vector between the point and the line:

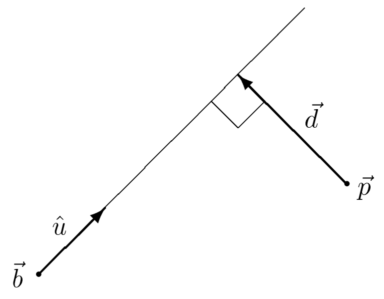

Fig. 3. Deviation, $\vec{d}$ of a point from a line, to be minimized through least squares to find best overlap of multiple lines of sight to the target point.

$$\begin{aligned}
\vec{d} &= \vec{b} + [(\vec{p} - \vec{b}) \cdot \hat{u}]\hat{u} - \vec{p} \\
d_j &= b_j + (p_0 - b_0)u_0u_j + (p_1 - b_1)u_1u_j \\
&\quad + (p_2 - b_2)u_2u_j - p_j
\end{aligned}$$

Taking the derivative of $\vec{d}$ with respect to each component of $\vec{p}$ yields

$$\begin{aligned}
\frac{\partial}{\partial p_0}\vec{d}^2 &= \frac{\partial}{\partial p_0}(d_0^2 + d_1^2 + d_2^2) \\
&= 2d_0\frac{\partial}{\partial p_0}d_0 + 2d_1\frac{\partial}{\partial p_0}d_1 + 2d_2\frac{\partial}{\partial p_0}d_2 \\
&= 2d_0(u_0^2 - 1) + 2d_1u_1u_0 + 2d_2u_2u_0 \\
\frac{\partial}{\partial p_1}\vec{d}^2 &= 2d_0u_0u_1 + 2d_1(u_1^2 - 1) + 2d_2u_2u_1 \\
\frac{\partial}{\partial p_2}\vec{d}^2 &= 2d_0u_0u_2 + 2d_1u_1u_2 + 2d_2(u_2^2 - 1)
\end{aligned}$$

We find the minimum deviation where the derivatives of the sums are zero:

$$\begin{aligned}\frac{\partial}{\partial p_0}\sum_i \vec{d}_i^2 &= \textstyle\sum_i \frac{\partial}{\partial p_0}\vec{d}_i^2 &= 0\\ &\quad \textstyle\sum_i \frac{\partial}{\partial p_1}\vec{d}_i^2 &= 0\\ &\quad \textstyle\sum_i \frac{\partial}{\partial p_2}\vec{d}_i^2 &= 0\end{aligned}$$

This can be recast into a form appropriate for matrix representation:

$$\begin{aligned}\sum_i c_{i00}p_0 + \sum_i c_{i01}p_1 + \sum_i c_{i02}p_2 + \sum_i e_{i0} &= 0\\ \sum_i c_{i10}p_0 + \sum_i c_{i11}p_1 + \sum_i c_{i12}p_2 + \sum_i e_{i1} &= 0\\ \sum_i c_{i20}p_0 + \sum_i c_{i21}p_1 + \sum_i c_{i22}p_2 + \sum_i e_{i2} &= 0\end{aligned}$$

Solve these 3 equations for the 3 unknowns: p_0, p_1, p_2:

$$\mathbf{C}\vec{p} = -\vec{e} \qquad \Rightarrow \qquad \vec{p} = -\mathbf{C}^{-1}\vec{e}$$

where the terms for $\mathbf{C}$ and $\vec{e}$ are defined by expanding and judiciously arranging the derivatives:

$$\begin{aligned}\frac{1}{2}\frac{\partial}{\partial p_0}\vec{d}^2 &= p_0(u_0^4 + u_0^2u_1^2 + u_0^2u_2^2 - 2u_0^2 + 1) +\\ &\quad p_1(u_1u_0^3 + u_1^3u_0 + u_1u_2^2u_0 - 2u_1u_0) +\\ &\quad p_2(u_2u_0^3 + u_2^3u_0 + u_2u_1^2u_0 - 2u_2u_0) +\\ &\quad b_0(-u_0^4 - u_0^2u_1^2 - u_0^2u_2^2 + 2u_0^2 - 1) +\\ &\quad b_1(-u_1u_0^3 - u_1^3u_0 - u_1u_2^2u_0 + 2u_1u_0) +\\ &\quad b_2(-u_2u_0^3 - u_2^3u_0 - u_2u_1^2u_0 + 2u_2u_0)\\ &\equiv c_{00}p_0 + c_{01}p_1 + c_{02}p_2 + e_0\end{aligned}$$

$$\begin{aligned}\frac{1}{2}\frac{\partial}{\partial p_1}\vec{d}^2 &= p_0(u_0^3u_1 + u_0u_1^3 + u_0u_2^2u_1 - 2u_0u_1) +\\ &\quad p_1(u_1^4 + u_1^2u_0^2 + u_1^2u_2^2 - 2u_1^2 + 1) +\\ &\quad p_2(u_2^3u_1 + u_2u_1^3 + u_2u_0^2u_1 - 2u_2u_1) +\\ &\quad b_0(-u_0u_1^3 - u_0^3u_1 - u_0u_2^2u_1 + 2u_0u_1) +\\ &\quad b_1(-u_1^4 - u_1^2u_0^2 - u_1^2u_2^2 + 2u_1^2 - 1) +\\ &\quad b_2(-u_2u_1^3 - u_2^3u_1 - u_2u_0^2u_1 + 2u_2u_1)\\ &\equiv c_{10}p_0 + c_{11}p_1 + c_{12}p_2 + e_1\end{aligned}$$

$$\begin{aligned}\frac{1}{2}\frac{\partial}{\partial p_2}\vec{d}^2 &= p_0(u_0^3u_2 + u_0u_2^3 + u_0u_1^2u_2 - 2u_0u_2) +\\ &\quad p_1(u_1^3u_2 + u_1u_2^3 + u_1u_0^2u_2 - 2u_1u_2) +\\ &\quad p_2(u_2^4 + u_2^2u_0^2 + u_2^2u_1^2 - 2u_2^2 + 1) +\\ &\quad b_0(-u_0u_2^3 - u_0^3u_2 - u_0u_1^2u_2 + 2u_0u_2) +\\ &\quad b_1(-u_1u_2^3 - u_1^3u_2 - u_1u_0^2u_2 + 2u_1u_2) +\\ &\quad b_2(-u_2^4 - u_2^2u_0^2 - u_2^2u_1^2 + 2u_2^2 - 1)\\ &\equiv c_{20}p_0 + c_{21}p_1 + c_{22}p_2 + e_2\end{aligned}$$

The above results in equations that produce a uniformly weighted result. If different weights are desired, replace each c_{ijk} and e_{ij} with w_ic_{ijk} and w_ie_{ij}, respectively. Weights based on the inverse-squared distance to each camera were used in the tracker by first computing the uniformly weighted solution to get approximate distances to the point and then returning to perform the weighted solution.

ACKNOWLEDGMENTS

The author would like to thank Jim Lloyd, Edward Tunstel, Ken Galbraith, and Eric Nieman for help fabricating the tracker; John Leimgruber for writing software to control the Leica Total Station used for surveying landmarks; Bruce Bon for his algorithm to find the closest crossover point of two lines; Peter Unold for his implementation of the permutation generator [7]; Andrew E. Johnson for his implementation of the quaternion method of Hebert [8]; Donald Gennery for all his work on the camera-model mathematics; and Mark Maimone and Jeff Biesiadecki for much encouragement and for valuable comments borne out of many hours of use.

REFERENCES

[1] J. N. Maki, J. F. Bell, K. E. Herkenhoff, S. W. Squyres, A. Kiely, M. Klimesh, M. Schwochert, T. Litwin, R. Willson, A. Johnson, M. Maimone, E. Baumgartner, A. Collins, M. Wadsworth, S. T. Elliot, A. Dingizian, D. Brown, E. C. Hagerott, L. Scherr, R. Deen, D. Alexander, J. Lorre, "The Mars Exploration Rover Engineering Cameras," Journal of Geophysical Research — Planets, Special Issue: The Mars Exploration Rover Mission, 2003.

[2] S. B. Goldberg, M. W. Maimone, and L. M. Matthies, "Stereo Vision and Rover Navigation Software for Planetary Exploration," 2002 IEEE Aerospace Conference proceedings, volume 5, Big Sky, Montana, USA, pp. 2025–2036, March 2002.

[3] J. J. Biesiadecki, M. W. Maimone, and J. C. Morrison. "The Athena SDM Rover: A Testbed for Mars Rover Mobility," iSAIRAS conference proceedings, Montreal, Canada, June 2001.

[4] L. M. Matthies, E. Gat, R. Harrison, B. H. Wilcox, R. A. Volpe, and T. E. Litwin, "Mars Microrover Navigation: Performance Evaluation and Enhancement," IEEE Conference on Robots and Systems (IROS), Pittsburgh, PA, Aug 5–9, 1995.

[5] L. M. Matthies, T. E. Litwin, and A. Kelly, "Obstacle Detection for Unmanned Ground Vehicles: A Progress Report," International Symposium of Robotics Research, Munich, Germany, October 1995.

[6] D. H. Ballard and C. M. Brown, *Computer Vision*, Englewood Cliffs, NJ: Prentice-Hall, 1982, ISBN 0–13–165316–4, p. 151.

[7] N. Dershowitz, "A Simplified Loop-Free Algorithm for Generating Permutations," BIT–15, pp. 158–164, 1975.

[8] O. D. Faugeras and M. Hebert, "The Representation, Recognition, and Locating of 3-D Objects," The International Journal of Robotics Research, Vol. 5, No. 3, Fall 1986.

[9] Y. Yakimovsky and R. T. Cunningham, "A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras," Computer Graphics and Image Processing 7, pp. 195–210, 1978.

[10] D. B. Gennery, T. E. Litwin, B. H. Wilcox, and B. B. Bon, "Sensing and Perception Research at JPL," Proc. IEEE Int. Conf. on Robotics and Automation, Raleigh, NC, pp. 311–317, March 31 – April 3, 1987.

[11] D. Gennery, "Least-Squares Camera Calibration Including Lens Distortion and Automatic Editing of Calibration Points," in *Calibration and Orientation of Cameras in Computer Vision*, A. Grun and T. Huang, editors, Springer-Verlag, ISBN 3–540–65283–3, 2001.

[12] D. B. Gennery, "Generalized Camera Calibration Including Fish-Eye Lenses," JPL document clearance number 03–0869. To be published, 2003.

[13] J. Crisp, T. E. Litwin, J. J. Lorre, D. A. Alexander, A. J. Runkle, T. J. Parker, and H. J. Moore, "Pathfinder rover cameras: resolution, stereometry, and geology," Abstract in Eos Trans. AGU, 78(46), Suppl: F403, Fall 1997.

[14] D. B. Gennery, "Visual Tracking of Known Three-Dimensional Objects," International Journal of Computer Vision, 7:3, pp. 243–270, 1992.

[15] B. H. Wilcox, K. Tso, T. E. Litwin, S. Hayati, B. B. Bon: "Autonomous Sensor Based Dual-Arm Satellite Grappling," NASA Conference on Space Telerobotics, Pasadena, CA, Jan. 31 – Feb. 2, 1989.

[16] D. B. Gennery, "Stereo Vision for the Acquisition and Tracking of Moving Three-Dimensional Objects," in *Techniques for 3-D Machine Perception*, North Holland: Elsevier Science Publishers B. V., 1986.

Todd Litwin started work at JPL in 1979, at first writing ground software to support navigating spacecraft. In the 1980s he moved into robotics research, concentrating on camera calibration, object tracking, and stereo ranging. In the late 1990s he started writing flight software for unmanned space projects, still keeping active in robotic vision. His camera-calibration and range-mapping software were used for the 1997 Mars Pathfinder project. His work on the Mars Exploration Rover project extends to flight software to acquire and process images at Mars from the flight cameras, and to manage data storage in non-volatile memory.